

# Selbststudium OOP6 & ALG1

## Auftrag

### Kapitel 5.2

1. zu bearbeitende Aufgabe: 5.1

*done*

2. Auf den Seiten 157/158 wird in der Methode start() ein while-loop verwendet.

Kreieren Sie ein Code-Fragment mit derselben Funktionalität unter Verwendung einer `do {...} while` Anweisung.

```
do
{
    // do something
} while (!finished)
```

### Kapitel 5.3

3. zu bearbeitende Aufgaben: 5.2 bis 5.5 sowie 5.7 bis 5.11

5.2:

- *Allgemeine Beschreibung*
- *Field Summary*
- *Constructor Summary*
- *Method Summary*
- *Field Detail*
- *Constructor Detail*
- *Method Detail*

5.3:

*startsWith(String prefix)*

*Überprüft ob ein String mit einer bestimmten Zeichenkette (prefix) startet.*

*startsWith(String prefix, int toffset)*

*Überprüft ob ein String ab einer bestimmten Position (toffset) mit einer bestimmten Zeichenkette (prefix) startet.*

5.4:

*Ja, public boolean endsWith(String suffix)*

*Parameter ist ein String (suffix), Returnwert ist Boolean (true/false)*

5.5:

*Ja, public int length()*

*Keine Parameter, Returnwert ist ein Integer mit der Anzahl Zeichen.*

5.7:

*public String trim()*

```
String text = "Hallo Du, das ist ein Test  "
String trimmedText = text.trim();
```

5.8:

*Innerhalb der Methode start() muss folgende Code-Zeile eingefügt werden:*

```
input = input.trim();
```

5.9:

*Innerhalb der Methode start() muss folgende Code-Zeile eingefügt werden:*

```
input = input.toLowerCase();
```

5.10:

*boolean*

5.11:

```
if(input.equals("bye")) {  
    finished = true;  
}
```

4. In welchem Package vermuten Sie die Klasse FileWriter?

Überprüfen Sie Ihre Vermutung mit Hilfe der Java API Dokumentation.

*java.io*

<http://docs.oracle.com/javase/1.4.2/docs/api/java/io/FileWriter.html>

5. Mit Hilfe der Klasse BufferedReader können Sie zeilenweise Files einlesen. Wie geht das? Die Antwort finden Sie wiederum in der Java API Dokumentation.

<http://docs.oracle.com/javase/1.5.0/docs/api/java/io/BufferedReader.html>

```
BufferedReader br = new BufferedReader(new FileReader("text.txt"));  
while ((thisLine = br.readLine()) != null)  
{  
    System.out.println(thisLine);  
}
```

## Kapitel 5.4

6. zu bearbeitende Aufgaben: 5.12 und 5.13

5.12:

<http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Random.html>

*Im Package: java.util*

*Es werden (pseudo) Zufallszahlen erzeugt.*

*Constructor: Random random = Random(); oder Random random = Random(seed);*

*Random Integer Zahl: random.nextInt();*

5.13:

```
import java.util.Random;
```

```
[...]
```

```
Random randomGenerator = new Random();  
// nextInt(100) gibt eine Zahl von 0 bis 99, deshalb +1 => Zahl von 1 bis 100  
int randomInt = randomGenerator.nextInt(100) + 1;
```

7. zu bearbeitende Aufgabe: 5.15

*random.nextInt(100) // Zufallszahl zwischen 0 bis und mit 99*

8. zu bearbeitende Aufgabe: 5.18

```
private ArrayList<String> responses;  
  
public RandomTester() // constructor  
{  
    responses = new ArrayList<String> ();  
    responses.add("yes");  
    responses.add("don't know");  
    [...]  
}  
  
public String getResponse()  
{  
    return responses.get(random.nextInt(responses.size()));  
}
```

## Kapitel 5.5

9. zu bearbeitende Aufgaben: 5.21 und 5.22

5.21:

```
import java.util.ArrayList;
import java.util.Random;

/**
 * The responder class represents a response generator object.
 * It is used to generate an automatic response to an input string.
 */
public class Responder
{
    private Random randomGenerator;
    private ArrayList<String> responses;

    /**
     * Constructs a Responder
     */
    public Responder()
    {
        randomGenerator = new Random();
        responses = new ArrayList<String>();
        generateResponses();
    }

    /**
     * Generate a response.
     * @return A string that should be displayed as the response
     */
    public String generateResponse()
    {
        return responses.get((randomGenerator.nextInt(responses.size())));
    }

    private void generateResponses()
    {
        responses.add("yes");
        responses.add("don't know");
        //etc...
    }
}
```

5.22:

*Die Länge der ArrayList passt sich automatisch an. Da es zusammen mit size() aufgerufen wird gibt es auch keine exception.*

## Algorithmen

10. Beschreiben Sie in Prosa oder in Pseudocode je einen Algorithmus für folgende Problemstellungen:

a) das Produkt von zwei ganze Zahlen berechnen (ein Multiplikations-Operator stehe nicht zur Verfügung!)

```
wenn a = 0
  dann return b
sonst solange b ≠ 0
  wenn a > b
    dann a = a - b
  sonst b = b - a
return a
```

b) die kleinste Zahl aus einer Folge bestimmen, z.B. -2 aus [4, -1, 50, 10, 0, 1, -2, 5, 10]

```
a = erstes Element aus der Liste
für alle Elemente b in der Liste
  wenn b < a
    dann a = b
return a
```

11. Implementieren Sie den Algorithmus zur Berechnung des grössten gemeinsamen Teilers (ggT) mit Hilfe des Modulo-Operators gemäss ALG1 Folie 12.

a) Testen Sie den den Algorithmus mit einigen Beispielen.

```
public int ggt(int num1, int num2)
{
  while((num1 != 0) && (num2 != 0)) {
    if(num1 > num2){
      num1 = num1 % num2;
    }
    else {
      num2 = num2 % num1;
    }
  }
  return (num1 + num2);
}
```

b) Sie dürfen nun den Modulo-Operator nicht mehr verwenden. Ersetzen Sie die Zeile mit den Modulo-Operatoren durch eine zusammengesetzte-Anweisung und testen Sie Ihre Implementierung erneut.

Hinweis: Die Modulo-Operation kann durch eine Folge von Subtraktionen ersetzt werden, welche beendet ist, wenn das Ergebnis der Subtraktion kleiner ist als der Subtrahend

```
/**
 * ggt Berechnung
 *
 * @param x Zahl 1
 * @param y Zahl 2
 * @return ggT von beiden Zahlen
 */
private int calcGGT(int x, int y)
{
  while (x != y)
  {
    if (x > y)
    {
      x = x - y;
    }
    else
    {
      y = y - x;
    }
  }
  return x;
}

/**
 * ggt Berechnen und ausgeben
 * @param x Zahl 1
 * @param y Zahl 2
 */
public void getGGT(int x, int y)
{
  System.out.println("Der ggT von " + x + " und " + y + " ist: " + calcGGT(x, y));
}
```

12. Welche Ordnung hat der Algorithmus zum Berechnen der n-ten Pseudo-Zufallszahl z? (siehe OOP6 Folie 28 und ALG1 Folie 15 ff.)

$$z_{n+1} = (a * z_n + r) \% m$$

$O(n)$

## Programmierübung

Schreiben Sie eine Klasse FormLetter, mit der Serienbriefe erstellt werden können.

Die Klasse besitzt eine ArrayList für Klassen des Typs FormalAddress (siehe OOP3).

Mit der Methode addAddress(...) wird ein neues Objekt vom Typ FormalAddress erzeugt und in die ArrayList hinzugefügt.

Mit der Methode

```
void print(int actualYear, String subject, String textBody)
```

wird für alle Elemente aus der ArrayList der Brief, bestehend aus Anrede, Subjektzeile und Briefftext, auf die Konsole ausgegeben.

```
import java.util.ArrayList;

/**
 * Write a description of class FormLetter here.
 *
 * @author Felix Rohrer (felix.rohrer@stud.hslu.ch)
 * @version 2013-03-24
 */
public class FormLetter
{
    private ArrayList<FormalAddress> fAddr;

    public FormLetter()
    {
        fAddr = new ArrayList<FormalAddress>();
    }

    public void AddAddress(FormalAddress newAddr)
    {
        fAddr.add(newAddr);
    }

    public void print(int actualYear, String subject, String textBody)
    {
        for (int i = 0; i < fAddr.size(); i++)
        {
            fAddr.get(i).createFormalFormOfAddress(actualYear);

            System.out.println(subject + "\n");
            System.out.println(textBody + "\n\n");
        }
    }
}
```